МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

по курсу «Основы микропроцессорной техники» (часть 2)
для студентов специальности 7.091302
«Метрология и измерительная техника»
дневной и заочной форм обучения

Утверждено
редакционно-издательским
советом университета,
протокол № 2 от 14.05.03

Харьков НТУ «ХПИ» 2003

Методичні вказівки до лабораторних робіт з курсу «Основи мікропроцесорної техніки» (частина 2), для студентів спеціальності 7.091302 «Метрологія та вимірювальна техніка» денної та заочної форм навчання / Уклад. О.І. Овчаренко, В.В. Лисенко, Р.П. Мигущенко, Л.О. Медведєва – Харків: НТУ «ХПІ», 2003.– 56 с.– Рос. мовою

Укладачі: О.І. Овчаренко,
В.В. Лисенко,
Р.П. Мигущенко,
Л.О. Медведєва

Рецензент В.І. Дякін

Кафедра вимірювально-інформаційної техніки

Лабораторная работа № 8

ИЗУЧЕНИЕ СТРУКТУРЫ И ПРАВИЛ ПОЛЬЗОВАНИЯ ДИРЕКТИВАМИ ПРОГРАММЫ МОНИТОР

Цель работы: изучить порядок запуска программы Монитор; изучить директивы Монитора; изучить правила пользования директивами Монитора; приобрести практические навыки в использовании директив Монитора.

Опыт 1. Запуск программы Монитор

Начальный адрес программы Монитор E000H. Запуск осуществляется путем набора начального адреса на клавишном регистре адреса панели контроля и отладки (ПКО) и ввода его с панели.

Порядок выполнения работы

1. Включить питание рабочего места;
2. Все клавиши ПКО привести в исходное (отжатое) положение;
3. Нажать клавишу РАБ/ОСТ панели контроля и отладки;
4. Набрать на клавишном регистре адреса ПКО адрес программы Монитор – E000H;
5. Последовательно нажать клавиши ПУСК, ВНА;
6. Отжать клавишу РАБ/ОСТ;
7. Нажать клавишу ПРД. В ответ на это на экране видеотерминала должно появится сообщение: МОНИТОР КТС ЛИУС-2. ВЕРСИЯ 4.0. После этого сообщения можно приступать к работе с программой Монитор;
8. При отсутствии сообщения (п.7) на экране видеотерминала – повторить опыт 1, начиная с п.2. Если сообщение отсутствует и при повторном проведении опыта, необходимо обратится к преподавателю.

Опыт 2. Изучение структуры программы Монитор

Программа Монитор является командно-управляемым операционным супервизором, который принимает команды оператора и управляет работой вычислительной системы. Монитор связан с оператором при помощи консоли (устройство отображения информации на ЭЛТ и клавиатуре). Диалог между оператором и Монитором осуществляется

при помощи команд (директив), набираемых и вводимых с клавиатуры, и ответных сообщений в виде блок-схем, показанных на рисунке 1.

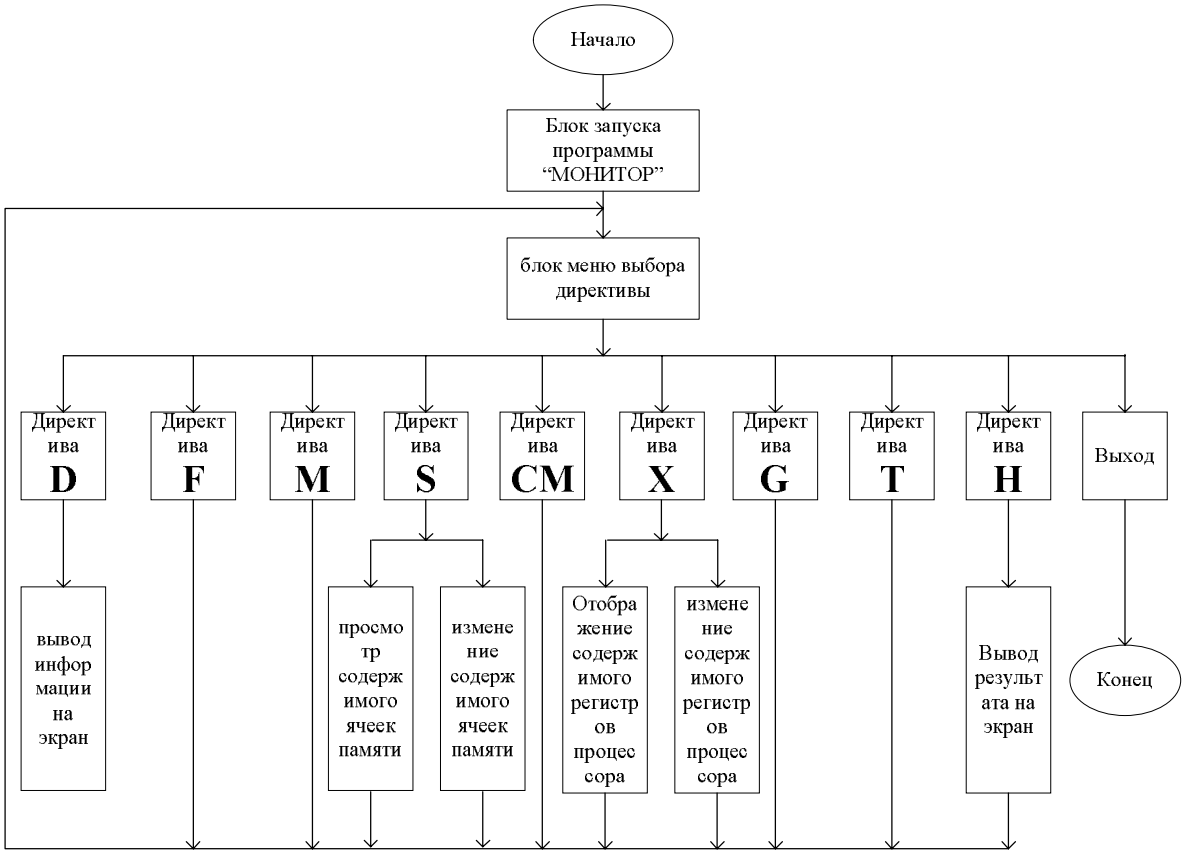


Рисунок 1 – Блок-схема взаимодействия программы Монитор и оператора.

Из указанной схемы следует:

1. После запуска программы Монитор пользователь попадает в блок-меню выбора директив;

2. Набором и вводом пользователь выполняет следующие функции:

– отображение на экране видеотерминала содержимого участка памяти (директива D);

– заполнение области памяти константой (директива F);

– пересылка содержимого памяти из одной области в другую (директива M);

– посредством директивы S выполняется:

а) просмотр содержимого ячеек памяти;

б) изменение содержимого ячеек памяти.

– сравнение содержимого двух областей памяти (директива CM);

– отображение на экране видеотерминала содержимого регистров процессора (директива X);

– изменение содержимого регистров процессора (директива XR, R – имя регистра);

– выполнение программы в автоматическом режиме (директива G);

– выполнение программы в пошаговом режиме (директива T);

– вычисление суммы и разности двух четырехзначных шестнадцатеричных чисел (директива H).

3. После выполнения одной из директив, необходимой пользователю, идет автоматический возврат в блок-меню выбора директивы;

4. Для выхода из программы Монитор необходимо выключить питание ЭВМ или перейти в программы Тест-Монитор или интерпретатор Бейсика.

Формат директив приведен в таблице 1.

Таблица 1 – Формат директив

№ п/п	Формат директив	Описание директив
1	D ADR1, ADR2 ВК	Отобразить содержание области памяти
2	F ADR1, ADR2, Y ВК	Заполнить область памяти константой Y, Y ∈ (00...FF)
3	M ADR1, ADR2, ADR ВК	Переслать содержимое памяти из одной области в другую
4	S ADR1 ВК (пробел)	Изменить (просмотр) содержимого ячеек памяти начиная с адреса ADR1
5	CM ADR1, ADR2, ADR3 ВК	Сравнить содержимое двух областей памяти
6	X ВК	Отобразить содержимое регистров процессора.
7	XR Y ВК	Записать в регистр R процессора число Y (в качестве R используются A, D, C, D, E, HL, SP, PC)
8	G ADR1, ADR01, ADR02 ВК	Начать выполнение программы в автоматическом режиме (ADR01, ADR02 – точки останова программы, задаются не обязательно)
9	T ВК	Выполнить программу в пошаговом режиме
10	H A(16) B(16)	Вычислить сумму и разность двух четырех-значных шестнадцатеричных чисел A и B

Примечание: 1. ADR1 – шестнадцатеричный адрес начала области памяти;
ADR2 – адрес конца области памяти;
ADR3 – адрес начала второй области памяти;
2. Максимальные значения адресов памяти определяются объёмом ОЗУ, для МикроДат составляет 7FFFH.

Порядок выполнения опыта

1. Набрать на клавиатуре директиву D, в соответствии с таблицей 1. Адреса ADR1 и ADR2, соответствующие границам области памяти, задает преподаватель. Ввести директиву нажатием клавиши ВК. Занести адреса и данные отображаемой области памяти в таблицу 2;

Таблица 2 – Адреса и данные отображаемой области памяти

Адрес	Данные

2. Набрать на клавиатуре директиву F, в соответствии с таблицей 1. Адреса ADR1, ADR2 и значение константы Y задает преподаватель. Ввести директиву нажатием клавиши ВК. Директивой D отобразить содержимое области памяти, в которую записывалась константа Y, и проверить правильность выполнения директивы;

3. Набрать на клавиатуре директиву M, в соответствии с таблицей 1. Значения ADR1, ADR2 взять из п.1 данного опыта, значение ADR3 приравнять значению ADR1 и п.2 данного опыта. Ввести директиву нажатием клавиши ВК. Директивой D отобразить содержимое области памяти, начиная с ADR3 и длиной (ADR2-ADR1)H.

Сравнить полученную информацию с данными таблицы 2;

4. Набрать на клавиатуре директиву S, в соответствии с таблицей 1. Значение ADR1 взять из п.1 данного опыта, нажимая несколько раз клавишу ПРОБЕЛ, просмотреть содержимое ячеек памяти и сравнить их с данными таблицы 2. Сообщения о данных в памяти разделяются символом "–".

5. Изменить содержимое памяти с начальным адресом ADR1 из п.4 с помощью директивы S. Для этого набрать директиву S, нажать клавишу ПРОБЕЛ, набрать новое содержимое ячейки и ввести его нажатием клавиши ВК. На экране появится сообщение об адресе и модифицированном содержимом ячейки памяти. В качестве нового содержимого ячеек памяти взять однобайтное шестнадцатеричное число;

6. Набрать на клавиатуре и ввести директиву X. При этом на экране появится сообщение о состоянии регистров процессора. В ответ на директиву X появится сообщение на экране видеотерминала:

A=AA B=BB C=CC E=EE D=DD H=12 L=34 M=1234 P=6789 S=7EC0

7. Изменить содержимое регистров микропроцессора. Для этой цели служит директива XR, в соответствии с таблицей 1. Здесь R – величина переменная. В качестве R используются A, B, C, D, E, H, L, M, P, S. Наименования регистров, содержимое которых нужно изменить, и данные, которые следует занести в эти регистры, задает преподаватель. Проверка выполнения директивы XR осуществляется директивой X;

8. Выполнить директиву H, в соответствии с таблицей 1. Значения чисел A(16) и B(16) задает преподаватель.

Содержание отчета

В отчете по лабораторной работе необходимо привести директивы Монитора и их описание, наименование регистров К580ИК80 и их обозначения.

Контрольные вопросы

1. Что такое Монитор?
2. Для чего предназначен Монитор?
3. Какова структура программы Монитор?
4. Как осуществляют запуск Монитора?
5. Для чего служат директивы D, F, M, S, X, XR, T, H, G?
6. Каков формат директив D, F, M, S, X, XR, T, H, G ?

Список литературы

1. Монитор. Руководство программиста 1582.02087-1
2. Григорьев В. П. Программное обеспечение микропроцессорных систем. М.: Энергоиздат, 1983. – С. 5 – 80
2. Коган Б. М., Сташин В. В. Микропроцессоры в цифровых сигналах. – М.: Энергия, 1979. – С. 26 – 33

Лабораторная работа № 9

РУЧНАЯ ТРАНСЛЯЦИЯ, ВВОД И ИСПОЛНЕНИЕ МАШИННЫХ КОМАНД

Цель работы: изучить систему команд микропроцессора К580ИК80; приобрести навыки ручной трансляции программ, написанных на языке Ассемблер микропроцессора К580ИК80; освоить процедуры ввода и выполнения основных машинных команд микропроцессора К580ИК80.

Опыт 1. Ручная трансляция программ написанных на языке Ассемблер

Совокупность машинных команд представляет собой базовый микропроцессорный язык. Для облегчения разработки программного обеспечения на основе базового процессорного языка создаются языки программирования более высокого уровня. Программы микропроцессорных систем должны выполнятся в минимальное время, занимать минимальный объем памяти и легко изменяться при необходимости. Всем этим условиям отвечает программирование на языке Ассемблер, оперирующем с мнемокодами и символьными адресами. Команды Ассемблера позволяют существенно упростить составление, чтение и отладку программ микропроцессорной системы. Каждая машинная команда обозначается символом, представляющим собой сокращенную форму полной записи наименования данной команды на алгоритмическом языке. Символ кодирования названия и содержания команд запоминается и воспринимается намного легче, чем двоичные или шестнадцатеричные коды.

При использовании Ассемблера необходимо учитывать специфические особенности конкретного микропроцессора, т.к. этот язык относится к машинно-ориентированным языкам программирования. Перевод программы, написанной на Ассемблере, в машинные коды называется трансляцией. Процесс трансляции сложных программ выполняется с помощью ЭВМ. Простые программы могут транслироваться вручную с привлечением таблиц. Команды Ассемблера микропроцессора К580ИК80 классифицируются следующим образом:

– команды пересылки;

– арифметические команды;

– логические команды;

– команды управления программой;

– специальные команды.

Ассемблер микропроцессора К580ИК80 имеет возможность работы с однобайтными и двухбайтными числами. Смысл трансляции заключается в переводе ассемблерной программы, выполненной в мнемокодах, в шестнадцатеричные коды.

Пример ручной трансляции программы:

Дан массив, состоящий из десяти чисел, расположенный в области ОЗУ начиная с адреса 1000Н. Переслать массив в область ОЗУ, начиная с адреса 1100Н. Программу пересылки расположить в памяти с адреса 0500Н. Блок-схема алгоритма выполнения данной программы приведена на рисунке 1.

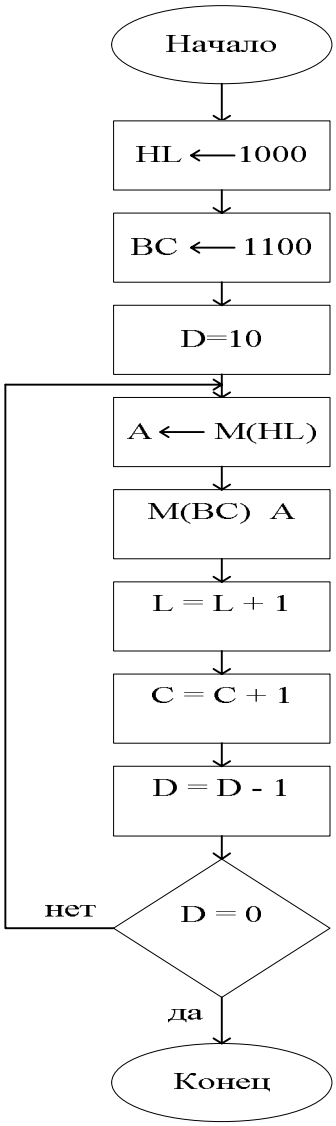


Рисунок 1 – Блок-схема программы

Программа

Адрес	Команда	Двоичный код	Код	Комментарии
0500	LXI HL	00100001	21	Загрузка регистровой пары HL
0501				адресом первого массива
0502				
0503	LXI BC	00000001	01	Загрузка регистровой пары BC
0504			00	адресом второго массива
0505			11	
0506	MVI D	00010110	16	Загрузка регистра D числом,
0507			0A	характеризующим величину массива
0508	MOV A, M(HL)	01111110	7E	Пересылка содержимого памяти адресуемой регистровой парой HL в аккумулятор
0509	STAX M(BC)	00000010	02	Пересылка содержимого аккумулятора в ячейку памяти, адресуемой регистровой парой BC
050A	INR L	00101100	2C	Увеличение содержимого регистров L
050B	INR C	00001100	0C	и C на единицу
050C	DCR D	000101010	15	Уменьшение содержимого регистра D на единицу
050D	JNZ		C2	Сравнение содержимого регистра D с
050E			01	нулем и переход по
050F			05	соответствующему адресу
0501	HLT		76	Останов

Порядок выполнения опыта

1. Получить от преподавателя задание (в соответствии с таблицей 1), значения адресов и данных;

2. Провести ручную трансляцию программы в машинные коды;

3. Транслированную программу показать преподавателю.

Таблица 1 – Варианты заданий для составления программы

№	Задание
1	Определить максимальный элемент массива
2	Определить минимальный элемент массива
3	Определить количество элементов массива больших 0F H
4	Определить количество элементов массива меньших 0F H
5	Определить младший байт суммы элементов массива больших 0F H
6	Определить младший байт суммы элементов массива меньших 0F H

Опыт 2. Изучение процедур ввода и выполнения машинных команд

Программа в машинных кодах должна быть введена в память микроЭВМ в данной лабораторной работе. Эта операция осуществляется с клавиатуры при помощи директивы S Монитора. Прогон программы

осуществляется в пошаговом режиме с поочередным применением директив X и T Монитора.

Порядок выполнения опыта

1. Ввести транслированную в опыте 1 программу в память, используя директиву S;

2. Проверить правильность ввода директивой D;

3. Отобразить содержимое регистров микропроцессора директивой X и результат занести в первую строку таблицы 2;

4. Директивой XP назначить начальный адрес выполняемой программы и отобразить состояние регистров директивой X. Результат занести во вторую строку таблицы 2;

5. Провести трассировку выполнения программы, последовательно применяя директивы T. Результаты трассировки записать в таблицу 2, начиная с третьей строки;

6. По окончанию трассировки результат показать преподавателю.

Таблица 2 – Результаты трассировки программы

№ строки	Адрес, команда, код	Состояния регистров										
		A	B	C	D	E	F	H	L	M	P	S
1	Состояния регистров											
2	Исходное состояние											
3												

Опыт 3. Изучение процедуры автоматического выполнения ассемблерной программы

Отладка и выполнение ассемблерной программы в пошаговом режиме не всегда удобна. В частности большие неудобства возникают при прогоне больших программ, циклических программ, программ, состоящих из множества коротких и отлаженных программ. В этих случаях целесообразно использовать автоматическое выполнение ассемблерных программ, применяя директиву G.

Порядок выполнения опыта

1. Отобразить на экране видеотерминала состояние программы, введенной в опыте 2. Для этого необходимо использовать директиву D.

При обнаружении ошибок в программе откорректировать ее, применяя директиву S;

2. Набрать директиву G в виде:

G ADR1 ВК,

где ADR1 – адрес начала ассемблерной программы. Если после выполнения директивы в таком формате на ПКО горит только индикатор ОЖ, то это значит, что программа выполнилась.

Для дальнейшей работы с ЭВМ необходимо перезагрузить программу Монитор. После чего проверить результат и сравнить с результатом опыта 2.

3. Набрать директиву G в виде:

G ADR1, ADR01, ADR02 ВК,

где ADR1 – адрес начала ассемблерной программы;

ADR01 – адрес промежуточного останова;

ADR02 – адрес конца программы.

После выполнения директивы ЭВМ остановится на адресе ADR01. Здесь можно проверить промежуточные результаты выполняемой программы.

Набрать G ВК – ЭВМ продолжит выполнение программы и остановится на адресе ADR02. После перезагрузки Монитора сравнить результаты с п.2 настоящего опыта.

Содержание отчета

Отчет по лабораторной работе представляет собой заполненную таблицу 2.

Контрольные вопросы

1. Что такое трансляция, как она осуществляется?

2. Перечислите типы команд микропроцессора К580ИК80.

3. Каково должно быть содержимое программного счетчика к моменту запуска программы?

4. Как изменяется содержимое программного счетчика при пошаговом выполнении программы?

5. Как организовать пересылку в микропроцессор данных длиной в два байта?

Список литературы

1. Григорьев В.А. Программное обеспечение микропроцессорных систем. М.: Энергоиздат, 1983. – С. 5 – 90.

2. Качян Б.М., Сташин В.В. Микропроцессоры в цифровых системах. М.: Энергия, 1979. – С . 26-3353 – 84.

3. Справочник по цифровой вычислительной технике. /Под ред. Б.Н. Малиновского. К.: Техника, 1981. – С. 22-24.

4. Монитор. Руководство программиста. 1582.02087-01.НПО САУ.

Лабораторная работа №10

ИЗУЧЕНИЕ ПРОЦЕДУР РАБОТЫ СО СТЕКОМ

Цель работы: изучение группы команд микропроцессора К580, относящихся к операциям со стеком; изучение принципов и практических приемов применения этих команд.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Стек – особый вид оперативной памяти, отличающийся последовательным способом организации доступа к ней.

Стек – это область памяти, в которой временно сохраняется информация, необходимая для осуществления возврата в программу после выполнения подпрограммы. Стек также используется для временного сохранения любых данных при нехватке внутренних регистров МП.

Стек играет огромную роль при обработке прерываний. Реагируя на внешние или внутренние прерывания, МП должен переключиться с текущей программы на другую программу, называемую обработчиком прерывания. Однако после обслуживания прерываний необходимо продолжить выполнения основной программы, для которой в стеке временно сохраняется содержимое используемых обработчиком прерывания регистров МП на время его работы.

Состояние стека отображается во внутреннем 16-битном регистре SP – указателе стека, который содержит адрес вершины стека.

В типовых МП фактическое положение стека в ОЗУ определяется программистом. До использования стека необходимо инициализировать SP

командой LXI SP на максимальный адрес области (вершину), выделяемой для стека.

Стек функционирует как память с последовательным доступом по типу данные, поступившие последними, извлекаются первыми (тип LIFO от LAST IN - FIRST OUT – последний входит - первый выходит или FILO от FIRST IN - LAST OUT – первый входит - последний выходит).

Содержимое любого регистра МП можно поместить в стек с помощью команды PUSH, а из стека можно извлечь командой POP последние включенные в него данные регистра МП. Обе стековые операции выполняются с 16-битовым словами, т.е. в стек можно включать только содержимое регистровых пар и извлекать из стека можно только регистровые пары.

Извлечение из стека не разрушает в памяти считываемые данные и, произведя специальными командами декремент SP на 2, можно вновь извлечь из стека ранее считываемые данные. При программировании надо следить за использованием стека, в частности, каждой команде PUSH должна соответствовать команда POP.

В МП 580 есть специфическая 1-байтная команда XTHG, которая производит обмен содержимого регистров H, L и двух верхних ячеек стека, т.е. последних загруженных в стек данных, при этом значение SP не изменяется.

К особенностям микропроцессора К580 относится однобайтная команда SPHL передачи 16-битных данных регистра HL в указатель стека.

Следует отметить, что команда CALL сочетает функции операций перехода и загрузки в стек. Адрес перехода к подпрограмме зафиксирован во втором и третьем байте команды CALL, а адрес следующей команды за CALL отправляется в стек. После завершения выполнения подпрограммы команда RET извлекает из стека адрес следующей команды основной программы и загружает его в счетчик команд PC.

Основным достоинством стека является то, что можно заносить в него данные (увеличивать емкость стека), не разрушая структуру уже записанных в нем данных. Если же данные запоминаются в ячейках памяти или в регистре, то теряется предыдущее содержимое этого участка памяти. Кроме того, МП может быстро передавать данные в стек и из стека, т.к. адрес содержится в указателе стека, а не является частью команды. Формат команд операций со стеком очень короткий.

Однако микропроцессор К580 не имеет никаких средств аппаратного контроля за поведением стека. Ошибки при работе со стеком приводят к двум ситуациям:

1. Переполнение стека – указатель стека достиг верхней границы области стека и делается попытка произвести дополнительное включение в стек;

2. Антипереполнение стека – указатель стека находится на нижней границе области стека и делается еще одна попытка извлечения из стека.

Возникновение любой из этих ситуаций ведет к тому, что поведение системы становится непредсказуемым.

Опыт 1. Изучение группы команд записи в стек

Микропроцессор К580 имеет в своей системе команд четыре команды записи в стек как показано в таблице 1.

Таблица 1 – Команды записи в стек

№	Мнемокод команд	Код команды	Выполняемые операции
1	PUSH BC	C5H	M<(sp)-1> ← (B); M<(sp)-2> ← (C); SP ← (SP)-2
2	PUSH DE	D5H	M<(sp)-1> ← (D); M<(sp)-2> ← (E); SP← (SP)-2
3	PUSH HL	E5H	M<(sp)-1> ← (H); M<(sp)-2> ← (L); SP← (SP)-2
4	PUSH PSW	F5H	M<(sp)-1> ← (A); M<(sp)-2> ← (F); SP← (SP)-2

Примечания:

1. В таблице 1 и далее запись (имя регистра) означает содержимое регистра с указанным именем.

Пример: (F) – содержимое регистра признаков F.

2. В таблице 1 и далее запись < > означает адрес.

Пример: M <(sp)-1> – ячейка памяти с адресом, равным исходному SP, уменьшенному на единицу.

Все команды таблицы 1 однотипны и осуществляют пересылку двухбайтного слова из регистровых пар микропроцессора в ОЗУ, адресуемое при этих операциях указателем стека SP. Специфической парой регистров является аккумулятор A и регистр признаков F, совместное содержимое которых называется словом состояния программы (PSW).

Важно понять и запомнить основные механизмы записи в стек:

– старший байт слова (соответственно содержимое регистров B, D, H, A для команд таблицы 1) пересылается в ОЗУ по старшему адресу

<(sp)-1>, где (sp) – содержимое указателя стека SP до выполнения команды;

– младший байт слова (соответственно содержимое регистров C, E, L, F для команд таблицы 1) пересылается в ОЗУ по младшему адресу <(sp)-2>, где (sp) – содержимое указателя стека SP до выполнения команды;

– по окончании выполнения команды содержимое указателя стека увеличивается на 2 по отношению к исходному содержимому.

Графическая интерпретация функционирования команд PUSH представлена на рисунке 1 (на примере команды PUSH BC).

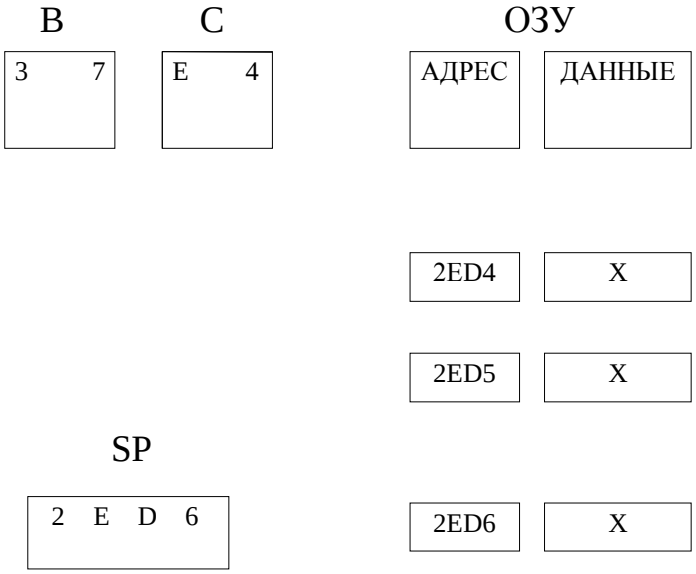


Рисунок 1. – Состояние МП и ОЗУ до выполнения команды PUSH BC:
X – произвольное содержимое

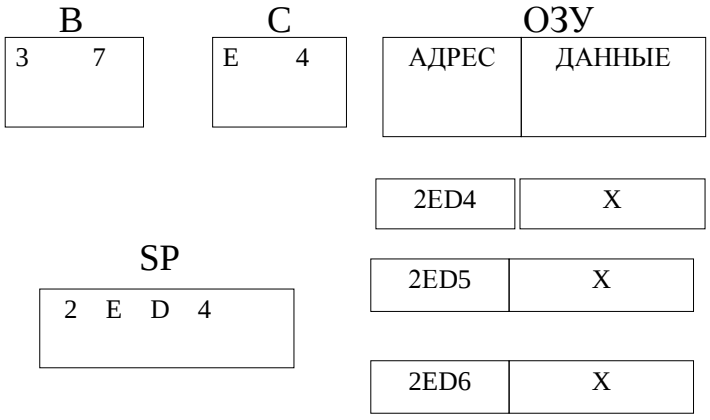


Рисунок 2. – Состояние МП и ОЗУ после выполнения команды PUSH BC:
X – произвольное содержимое

Как видно из рисунка 1 и рисунка 2 работа стека при записи напоминает укладку листов в пачку. При этом листы (байты) как бы укладываются парами поверх пачки, а указатель стека хранит номер (адрес) последнего из уложенных листов (байтов).

Порядок выполнения опыта

1. Включить рабочее место и запустить программу Монитор;

2. Выполнить директиву X Монитора. Обратить внимание на то, что всегда после запуска программы Монитор (SP) = 7EC H, (A) = AAH, (B) = BBH, (C) = CCH, (D) = DDH, (E) = EEH, (F) = FFH, (H) = 12H, (L) = 34H;

3. Директивой S ввести программу записи в стек содержимого регистров МП;

АДРЕС	ДАННЫЕ
1000	C5
1001	D5
1002	E5
1003	F5

4. Директивой XP установить содержимое программного счетчика равным 1000H;

5. Директивой S (или F) установить нулевое содержимое ячеек памяти по адресам 7EB8H...7EBFH;

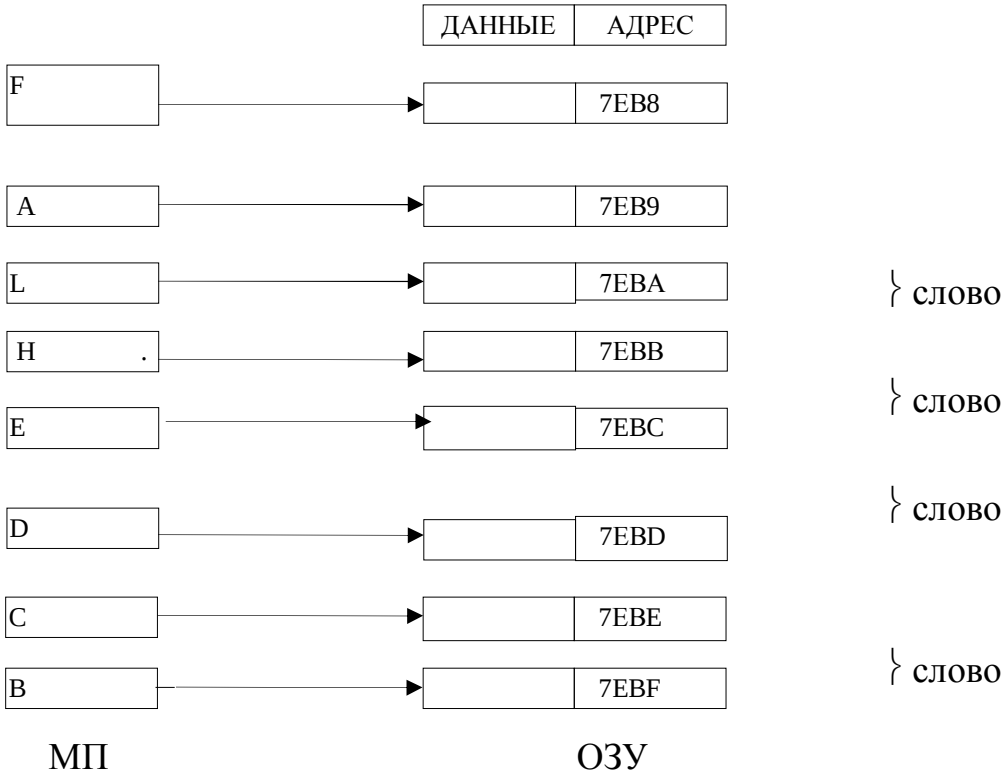
6. Отобразить директивой X состояние МП, выполнить с помощью директивы T одну команду программы п.3. Убедиться, что содержимое указателя стека уменьшилось на 2;

7. Отобразить директивой S содержимое ячеек ОЗУ с адресами <(SP) - 1>, <(SP) - 2>. Выяснить, содержимое каких регистров попало в эти ячейки;

8. Выполнить п.6, п.7 еще три раза для полного прогона программы п.4;

9. Отобразить содержимое восьми ячеек ОЗУ, начиная с адреса 7EB8H. Убедиться, что результат работы программы п.3 соответствует таблице 2.

Таблица 2 – Результат работы программы записи содержимого регистров МП в стек



Опыт 2. Изучение группы команд чтения из стека

Микропроцессор К580 имеет в своей системе команд четыре команды чтения из стека (таблица 3).

Таблица 3 – Команды чтения из стека

№	Мнемокод команд	Код команды	Выполняемые операции
1	POP BC	C1H	B ← (M<(sp)+1>); C ← (M<(sp)>); SP ← (SP)+2
2	POP DE	D1H	D ← (M<(sp)+1>); E ← (M<(sp)>); SP ← (SP)+2
3	POP HL	E1H	H ← (M<(sp)+1>); L ← (M<(sp)>); SP ← (SP)+2
4	POP PSW	F1H	A ← (M<(sp)+1>); F ← (M<(sp)>); SP ← (SP)+2

Все команды таблицы 3 однотипны и осуществляют пересылку двухбайтного слова из ОЗУ в регистровые пары МП. Механизмы чтения из стека таковы:

– старший байт слова в ОЗУ, расположенный по старшему адресу <(SP) + 1 >, передается в регистры B, D, H, A (в соответствии с командами таблицы 3);

– младший байт слова в ОЗУ, расположенный по младшему адресу <(SP)>, передается в регистры C, E, L, F (в соответствии с командами таблицы 3);

– по окончании выполнения команды содержимое указателя стека увеличивается на 2 по отношению к исходному состоянию.

Графическая интерпретация функционирования команд POP представлена на рисунке 3 и рисунке 4 (на примере команды POP PSW).

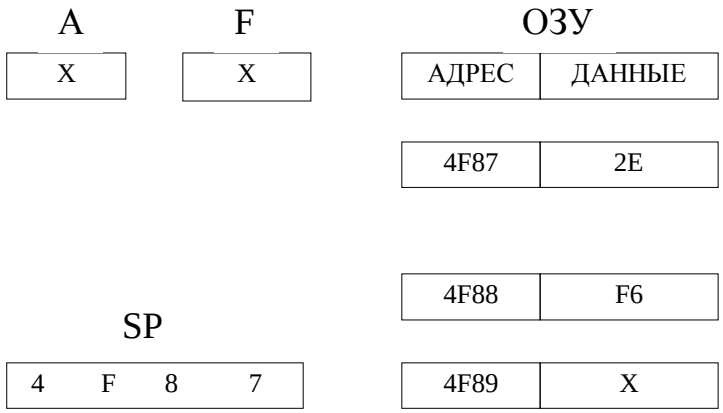


Рисунок 3. – Состояние МП и ОЗУ до выполнения команды POP PSW

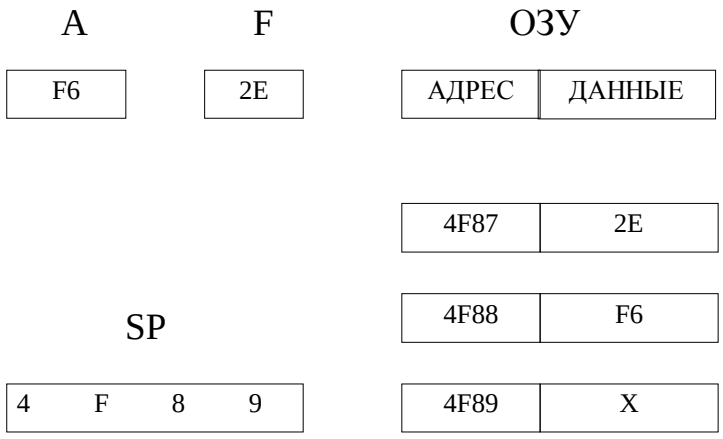


Рисунок 4. – Состояние МП и ОЗУ после выполнения команды POP PSW

Как видно из рисунка 3 и рисунка 4 работа со стеком при чтении напоминает извлечение пар листов из пачки. При этом листы отбираются с вершины пачки, а указатель стека хранит номер верхнего листа, оставшегося в пачке.

Совместный анализ рисунка 1, 2 и рисунка 3, 4 показывает, что работа на стеке осуществляется по принципу "первым вошел – последним вышел",

т.е. данные, которые при последовательном обращении к стеку записывались в него первыми, будут прочитаны последними.

Порядок выполнения опыта

1. Ввести программу чтения из стека:

```
1000  C1H
1001  D1H
1002  E1H
1003  F1H
```

2. Установить содержимое программного счетчика:

(PC) = 1000H

3. Записать из Монитора в ячейки ОЗУ с адресами 7EB8H...7EBFH такие данные, чтобы при чтении их в регистрах МП программой п.1 результат оказался бы равным (A) = 00H, (B) = 1H, (C) = 2H, (D) = 3H, (E) = 4H, (F) = 5H, (H) = 6H, (L) = 7H;

4. Выполнить программу п.1 и убедиться, что в регистрах МП находятся данные, оговоренные в п.3.

Опыт 3. Применение стека при буферировании данных

В ряде задач информационно-измерительной техники необходимо осуществить пакетную обработку информации, поступающей от исследуемого (измеряемого) объекта, например, через аналого-цифровой преобразователь. В этом случае процедура ввода и накопления данных в ОЗУ (буферирование) разделены во времени с процессом обработки этих данных. При больших объемах информации, подлежащей буферированию, и высоких скоростях ее ввода обычно используется режим прямого доступа к памяти. В рядовых случаях можно использовать транзитную передачу через процессор в ОЗУ. Такая передача, в частности, может использовать и операции со стеком.

Порядок выполнения опыта

1. Разработать программу ввода данных от 16-разрядного АЦП и буферирование их в ОЗУ с помощью операций со стеком. Блок-схема программа приведена на рисунке 3 и предполагает использование в качестве источников информации регистров адреса ПКО (младший регистр

– порт с адресом 1, старший регистр – порт с адресом 2), а также синхронный способ обмена (без подтверждения готовности);

2. Ввести в ОЗУ разрабатываемую программу;

3. Набрать на клавишных регистрах адреса ПКО отличающиеся друг от друга данные;

4. Выполнить два цикла программы, изменив в начале второго цикла набор на клавишных регистрах;

5. Отобразить введенную информацию на экран видеотерминала.

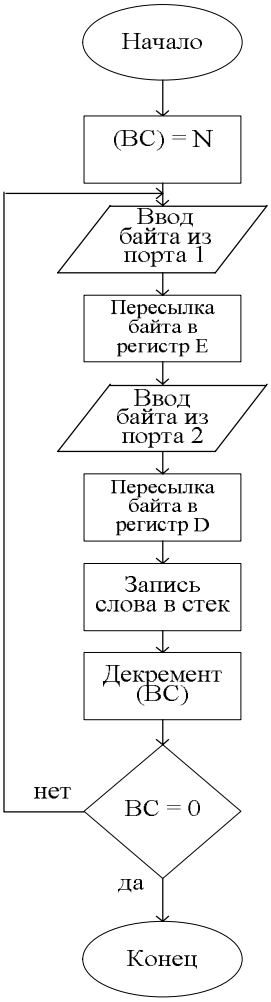


Рисунок 3 – Блок-схема ввода и буферизации данных

Контрольные вопросы

1. Что такое стек?

2. В какой области ОЗУ МикроДат может быть организован стек?

3. Каким образом может быть классифицирована стековая память с точки зрения доступа к ней?

4. Каковы принципы записи в стек и чтение из стека в системе команд М580?

5. Какой регистр МП хранит адрес "верхушки" стека?

6. Каким образом можно передавать содержимое регистров DE в регистр указателя стека SP?

Лабораторная работа №11

МОДЕЛИРОВАНИЕ ПРОЦЕДУР ВВОДА ИНФОРМАЦИИ В МИКРО-ЭВМ

Цель работы: приобретение практических навыков моделирования на средствах МикроДАТ процедур ввода квантованной информации.

Ввод информации каналов в микро-ЭВМ является одной из основных задач при построении информационно-управляющих систем. Важность этой задачи обусловлена тем, что от организации ввода в значительной мере зависят такие характеристики измерительных каналов, как частота дискретизации, динамическая погрешность. Сопутствующими для ввода задачами, как правило, оказываются задачи первичной обработки: преобразование форматов данных, определение знака входных данных, линеаризации характеристик первичных измерительных преобразователей, нормирование, функциональное преобразование и т.д.

Грамотное решение задач ввода и первичной обработки, порой позволяет получить лучшие результаты на "медленных" АЦП и процессорах, чем неграмотное решение на "быстрых" АЦП и процессорах.

Задачи ввода распадаются на два относительно самостоятельных класса по типу дискретных сигналов: одноканальные бинарные и многоканальные бинарные. Значительно реже встречается задача ввода сигналов от трехпозиционных датчиков (по типу полярного реле). Одноканальные бинарные сигналы (статические и динамические) характерны для задач программного логического управления, обработки информации импульсных датчиков (с частотной, фазовой и временной модуляцией). Многоканальные бинарные сигналы характерны для задач измерения и управления при наличии датчиков с непрерывным (аналоговым) выходом и подразумевают предварительные (до ввода) преобразования непрерывных сигналов в бинарные с помощью АЦП.

Опыт 1. Ввод одноканальных бинарных сигналов

Бинарный одноканальный сигнал, показанный на рисунке 1, может нести различную информацию.

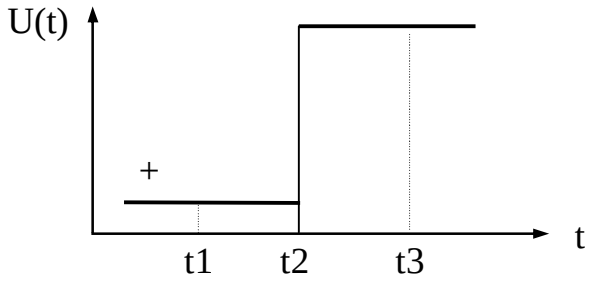


Рисунок 1 – Бинарный одноканальный сигнал

В статическом состоянии сигнала информацию несет состояние U(t) (логический "0" или логическая "1") в определенный момент времени (момент опроса). Так, при опросе датчика в момент времени t1 значение введенного байта будет равно "0", а при опросе в момент t3 – равно "1".

В данном случае информацию несет смена состояния бинарного сигнала с "0" на "1" (или наоборот). Здесь речь фактически о моменте t2 (рисунок 1).

Ввод бинарных сигналов в МП580 может быть осуществлен единственной командой IN. При этом для одноканальных сигналов, как правило, приходится выполнять либо выделение нужного бита (например, соответствующим числом сдвигов с итоговым размещением вводимого сигнала в признак CARRY), либо маскирование ненужных битов (например, логическим умножением на байт, у которого во всех битах кроме интересующего расположены нули). При вводе нескольких бинарных одноканальных сигналов с целью упрощения их последующей логической обработки целесообразно хранить их в виде байтов с информационным битом в одноименных позициях байтов (например, старший или младший).

Порядок выполнения опыта

1. Написать ассемблерную программу ввода бинарных одноканальных статических сигналов из портов с адресами 01, 02, 03 и их логической обработки. Расположение битов в байтах и требуемую логическую функцию задает преподаватель в соответствии с таблицей 1;

2. Ввести программу, отладить ее, результат показать преподавателю;

Таблица 1 –Варианты заданий расположения битов в портах

Вариант №	Положение бита в портах			Логическая функция
	Порт 01	Порт 02	Порт 03	Y = F(x)
1 2	0 2	1 3	2 6	Y = X1^X2^X3
3 4	7 5	2 1	4 3	Y = X1vX2vX3
5 6	2 1	7 3	1 5	Y = X1^X2^X3
7 8	2 7	0 4	6 3	Y = X1vX2vX3
9 10	6 3	4 4	2 5	Y = (X1^X2)vX3
11 12	4 0	5 2	6 4	Y = (X1vX2)^X3
13 14	1 3	2 0	0 1	Y = X1+X2+X3
15 16	6 7	5 6	1 0	Y = X1+X2+X3

3. Написать программу ввода динамического одноканального бинарного сигнала в соответствии с алгоритмом, показанным на рисунке 2. Вариант задания в соответствии с таблицей 2 получить у преподавателя;

4. Ввести программу, отладить ее, результат показать преподавателю;

5. Оценить максимальную погрешность при вводе динамического бинарного сигнала.

Таблица 2 – варианты заданий для программы

Вариант №	Адрес порта	Позиция бита	Тип перехода
1	1	0	“0” – “1”
2	2	1	“1” – “0”
3	3	2	“0” – “1”
4	4	3	“1” – “0”
5	5	4	“0” – “1”
6	6	5	“1” – “0”
7	7	6	“0” – “1”
8	8	7	“1” – “0”

Опыт 2. Ввод многоканального бинарного сигнала АЦП

Большинство АЦП имеют переменное время преобразования, зависящее от значения преобразуемого сигнала. В этой связи можно показать два возможных алгоритма ввода данных от АЦП. Синхронный алгоритм, показан на рисунке 3, предполагает, что максимальное время преобразования АЦП известно.

Алгоритм на рисунке 3 требует некоторых комментариев. Во-первых, расшифруем понятие "подготовка АЦП к преобразованию". С точки зрения выполняемых при этом функций логично отметить такие:

– сброс (установка) элементов памяти АЦП в исходное состояние;

– выбор канала (для многоканальных АЦП);

– настройка АЦП (выбор частоты квантования, опорных напряжений и т.д.).

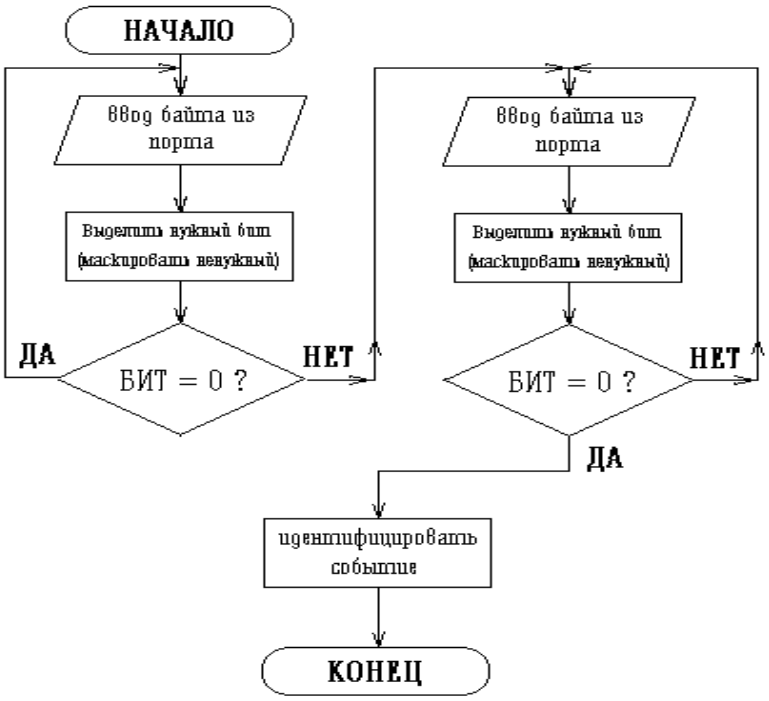


Рисунок 2 – Алгоритм ввода перехода “0” – “1”

Во-вторых, отметим, что временная задержка в синхронном алгоритме (рисунок 3) должна быть больше максимального времени преобразования АЦП. Наконец, сам термин "синхронный" обозначает, что данные вводятся без предварительного опроса готовности АЦП к обмену. Здесь речь по

существу идет о равномерной дискретизации непрерывного сигнала во времени, так как алгоритм (рисунок 3) не имеет ветвлений.

Асинхронный алгоритм, показанный на рисунке 4, отличается наличием блоков определения готовности АЦП к обмену. Комментарии по организации асинхронного обмена можно найти в [1]. Здесь же отметим, что необходимость в определении готовности приводит к переменному интервалу ввода данных и делает дискретную непрерывную сигнала неравномерной во времени.

Порядок выполнения опыта

1. Написать ассемблерную программу, реализующую синхронный алгоритм (рисунок 3). Данные о программе получить от преподавателя в соответствии с таблицей 3;

2. Временную задержку оформить подпрограммой. Для создания задержки использовать в цикле "пустые" операции (NOP; MOV R1, R1 и т.д.);

3. Ввести программу, отладить ее, результат показать преподавателю;

4. Модифицировать предыдущую программу в соответствии с асинхронным алгоритмом (рисунок 4). Дополнительные параметры программы получить у преподавателя в соответствии с таблицей 4;

5. Ввести программу, отладить ее, результат показать преподавателю.

Рисунок 3 – Синхронный алгоритм ввода данных от АЦП

Таблица 3 – Данные о программе

№ варианта	Величина задержки, S	Порт данных АЦП	Порт/байт подготовки АЦП	Порт/бит запуска АЦП
1	10	порт 01	02 - 04H	03 / X0 = 1
2	20	порт 02	03 - 08H	01 / X1 = 0
3	30	порт 03	01 - C0H	02 / X2 = 1
4	40	порт 01	02 - C4H	03 / X4 = 1
5	50	порт 02	03 - C8H	01 / X4 = 1
6	60	порт 03	01 - CCH	02 / X5 = 0
7	70	порт 01	02 - D0H	03 / X6 = 0
8	80	порт 02	03 - D4H	01 / X7 = 0

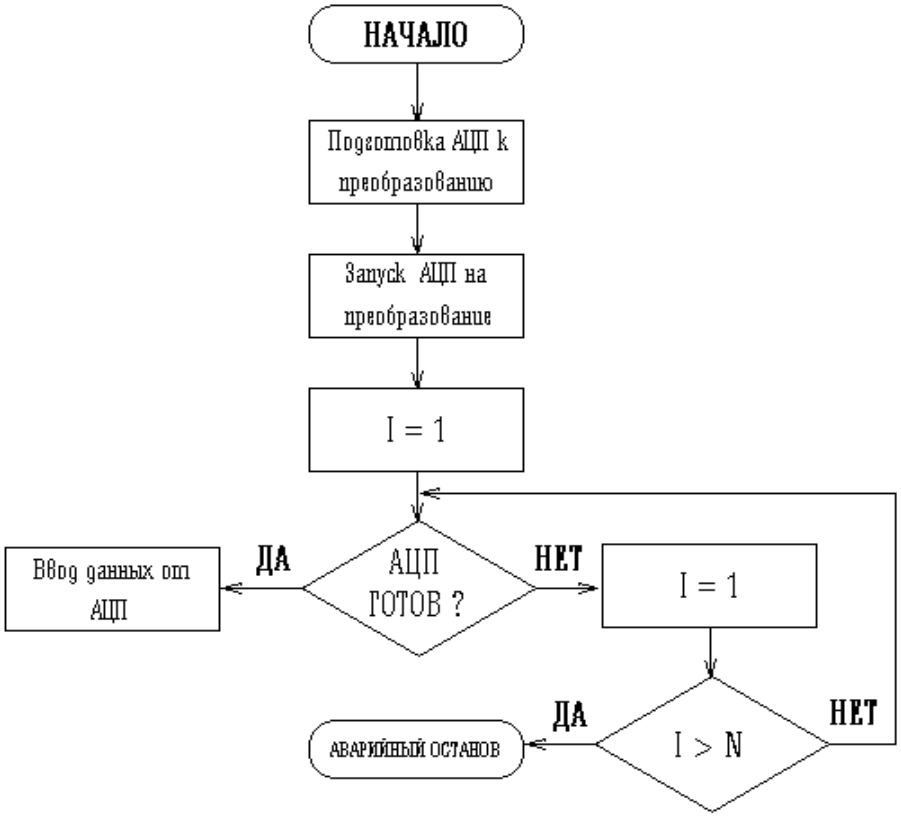


Рисунок 4 – Асинхронный алгоритм ввода данных от АЦП

Таблица 4 – Дополнительные параметры прграммы

№ варианта	Значение N(10)	Порт/бит готовности
1	10	03 / X7 = 1
2	20	01 / X6 = 0
3	30	02 / X5 = 0
4	40	03 / X4 = 0
5	50	01 / X3 = 0
6	60	02 / X2 = 0
7	70	03 / X1 = 1
8	80	01 / X2 = 0

Контрольные вопросы

1. Каковы источники погрешностей при вводе одноканальных статических и динамических бинарных сигналов?
2. Каковы принципиальные пути снижения погрешностей ввода одноканальных бинарных сигналов?
3. Каковы типичные технические задачи, требующие ввода бинарных одноканальных сигналов?
4. Каковы сравнительные преимущества и недостатки синхронных и асинхронных алгоритмов?
5. Чем определяется величина задержки в синхронном алгоритме?
6. Чем определяется параметр N в асинхронном алгоритме?
7. В чем состоят процедуры подготовки, пуска и ввода из АЦП?

Список литературы

1. Методические указания к лабораторным работам по курсу "Вычислительная техника в измерительных устройствах и микропроцессоры".

Лабораторная работа №12

ОБРАБОТКА МАССИВОВ ИНФОРМАЦИИ В МИКРО-ЭВМ

Цель работы: изучить блок-схемы обработки массивов числовой информации в микро-ЭВМ, приобрести практические навыки в использовании рациональных методов адресации при работе с массивами, рассмотреть типовые для информационно-измерительной техники примеры обработки массивов.

Опыт 1. Изучение простейших блок-схем обработки массивов числовой информации

В информационно-измерительной аппаратуре с применением микропроцессоров и микро-ЭВМ достаточно часто встречаются задачи обработки массивов измерительной информации. При этом в общем случае предполагается, что К-мерный массив уже введен тем или иным способом

в ОЗУ и необходимо осуществить его обработку по определенным алгоритмам. Такими задачами, например, могут быть: изменение знака чисел в одномерном массиве на противоположный (преобразование в дополнительный код), сортировка чисел по знаку и (или) по модулю, вычислительные операции с многомерными массивами (косвенные или совокупные измерения, например, косвенное определение мощности на постоянном или переменном токе), переформатирование (округление или усечении) данных и т.д. Характерной особенностью рассматриваемых задач является цикловая структура реализующих алгоритмов, причем для многомерных массивов зачастую присутствуют и вложенные циклы (цикл в цикле). Для одномерных массивов достаточно общей структуры обработки, которая иллюстрируется на рисунке 1.

Особенностью, непопадающей под обобщенный алгоритм (рисунок 1), является необходимость сохранить исходные данные. В терминах вычислительной техники эта ситуация требует раздельного хранения исходного и обрабатываемого массива в ОЗУ. Для этого случая блок-схема (рисунок 1) может быть модифицирована к виду, показанному на рисунке2.

Порядок выполнения опыта

1. Получить от преподавателя вариант задачи по обработке одномерного массива в соответствии с таблицей 1;
2. Разработать и оттранслировать ассемблерную программу;
3. Включить рабочее место и ввести программу;
4. Отладить программу;
5. Результаты обсудить с преподавателем.

Опыт 2. Изучение особых случаев обработки массивов

Оба приведенных в опыте 1 алгоритма предполагают, что длина массива в результате обработки сохраняется, т.е. другими словами один исходный массив превращается в другой преобразованный массив той же длины. Однако существуют и задачи, которые по своей постановке требуют ибо изменения (как правило, сокращение) длины массива, либо изменения количества массивов. Примером сокращения длины массива

при обработке может служить задача выборки положительных (отрицательных) чисел из исходного массива.

Таблица 1 – Варианты задачи обработки одномерного массива

№ варианта	Необходи-мость в сохранении исходного массива	Начальный адрес исходного массива	Длина массива в байтах	Начальный адрес обрабатыва-емого массива	Сущность алгоритма обработки
1	нет	4000H	28H	-	Изменить знак элемента массива на противоположный
2	да	3000H	16H	2000H	⟵ ‖ ⟵
3	нет	3600H	18H	-	Добавить к элементу массива (байта) аддитивное смещение 02H. При переполнении разрядной сетки результату присваивают значение FF¦
4	да	2750H	44H	1600H	⟵ ‖ ⟵
5	нет	1260H	12H	-	Сдвинуть байт на 2 бита вправо (разделить байт на 4)
6	да	3780H	32H	2400H	⟵ ‖ ⟵
7	нет	5100H	20H	-	Замаскировать старший байт
8	да	4150H	8H	1210H	⟵ ‖ ⟵
9	нет	6540H	10H	-	обнулить младшую тетраду байта
10	да	1700H	24H	5000H	⟵ ‖ ⟵

Примером изменения количества массивов может служить задача сортировки чисел на положительные и отрицательные (один исходный

массив превращается в два) и задача вычисления суммы элементов массивов (два исходных массива превращаются в один).

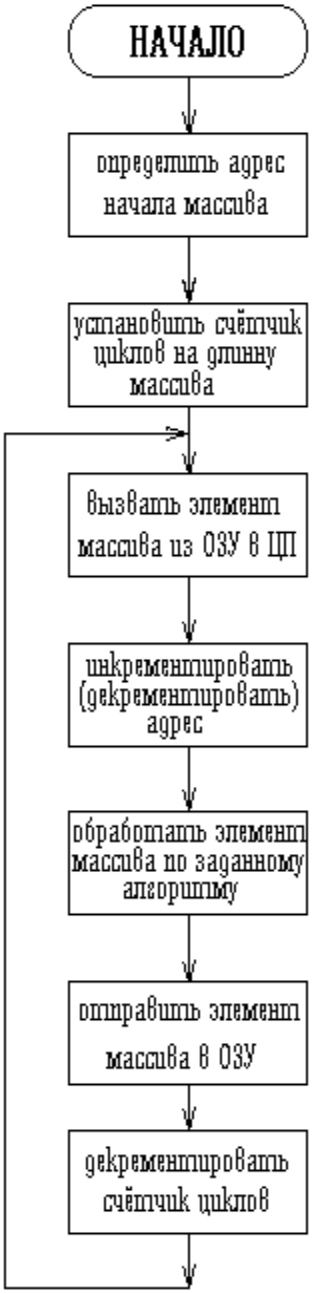


Рисунок 1 – Обобщённая блок-схема обработки одномерного массива

Порядок выполнения опыта

1. Получить от преподавателя вариант задачи в соответствии с таблицей 2;

2. Адреса и длины массивов выбрать самостоятельно;

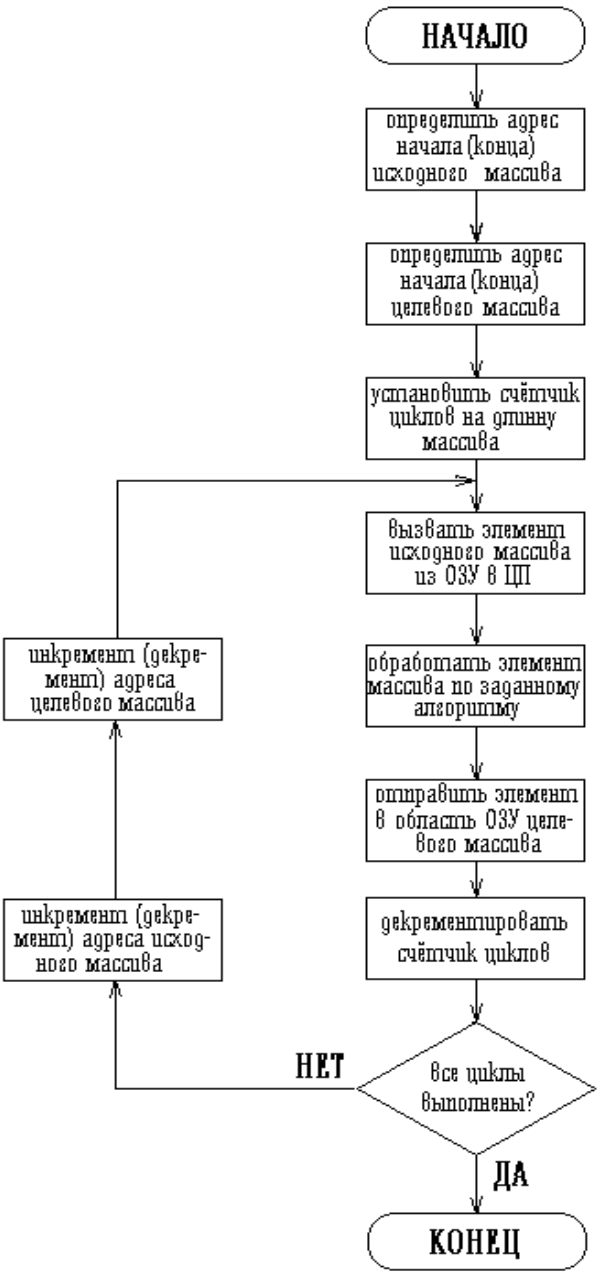


Рисунок 2 – Блок-схема обработки одномерного массива для случая, когда необходимо сохранение исходного массива

Таблица 2 – Варианты задач обработки массива с сохранением данных

№ варианта	Формулировка задач
1.	Из исходного массива отобрать положительные числа исходный массив сохранить в ОЗУ
2.	Из исходного массива отобрать отрицательные числа. Новый массив разместить, начиная с начального адреса исходного массива
3.	Рассортировать исходный массив по знаковому признаку
4.	Рассортировать исходный массив по признаку четности числа единиц в байте
5.	Сложить элементы двух исходных массивов. Переносами пренебречь
6.	Вычислить поразрядную конъюнкцию элементов двух массивов

Содержание отчета

В отчете о лабораторной работе необходимо привести блок-схемы и ассемблерные программы выполненных задач.

Контрольные вопросы

1. Какова характерная особенность структуры программы для обработки массивов?
2. Что такое длина и размерность массива?
3. Каковы типичные задачи, связанные с обработкой массивов?
4. Какие способы адресации выгодно использовать при обработке массивов?
5. Каковы качественные состояния между количеством и длиной исходных и обработанных массивов?
6. Каким образом при обработке массивов можно использовать индексную и относительную адресацию?
7. Чем ограничены размеры обрабатываемых массивов?
8. Каким образом необходимо обрабатывать массивы, элементы которых имеют большую длину, чем разрядная сетка процессора?

Лабораторная работа №13

ПРИМЕНЕНИЕ МИКРО-ЭВМ ДЛЯ МОДЕЛИРОВАНИЯ АППАРАТНЫХ СРЕДСТВ

Цель работы: овладение приемами создания программных моделей аппаратных средств (на примерах комбинационных логических устройств).

Необходимость в создании программных моделей аппаратных средств возникает либо в ходе проектирования аппаратного обеспечения, либо при перераспределении функций между аппаратными и программными средствами системы. Примером первой из указанных ситуаций может быть создание некоторой логической схемы. Для проверки правильности ее функционирования, выяснения возможного наличия "гонок" при прохождении сигналов и окончательной отладки

вовсе необязательно собирать макет данной схемы. Бывает целесообразным (особенно при наличии банка моделей) решить эти проблемы путем моделирования. Примером второй из указанных ситуаций является возложение на программную часть некоторой системы функций аппаратной части с целью минимизации последней. Такими функциями, в частности, могут быть: преобразование кодов, счет, формирование временных интервалов и задержки, формирование регулярных и нерегулярных импульсных последовательностей и т.д.

Таким образом, программные модели аппаратных средств являются достаточно мощным средством проектирования устройств и систем. Их использование может значительно сократить время разработки, отладки, тестирования и модификации устройств и систем, сократить аппаратные затраты при одновременном обеспечении гибкости средств, быстро и высокоэффективно исследовать все возможные режимы работы. Различают два способа построения модели: компиляционный и интерпретирующий.

При компиляционном способе для каждого объекта моделирования конструируется своя программа, которая для выполнения не требует никаких входных данных, кроме входного набора переменных (входного вектора).

При интерпретирующем способе конструируется достаточно универсальная программа, моделирующая работу группы или даже класса объектов. Работа с такой моделью, помимо входного вектора переменных требует и задания вектора настройки, т.е. некоторого массива данных, по которым осуществляется настройка программы на конкретный тип модели.

В данной работе изучаются приемы моделирования комбинационных логических устройств (схем без памяти).

Опыт 1. Моделирование интегральной схемы К155ЛР1

Интегральная схема К155ЛР1, как известно, реализует следующую функцию алгебры логики:

$$Y_1 = \overline{X_1 \wedge X_2 \wedge X_3 \wedge X_4} \qquad (1)$$

$$Y_2 = \overline{X_5 \wedge X_6 \wedge X_7 \wedge X_8} \ , \qquad (2)$$

где Y_i – входные переменные; i $\in$ 1...8;

Y_j – выходные переменные; j $\in$ 1,2.

Рассмотрим алгоритм моделирования функций (1), (2).

Пусть в некоторый регистр R1 процессора тем или иным образом введен входной вектор

$$(R_1) = X_8 X_7 X_6 X_5 X_4 X_3 X_2 X_1$$

Сдвинем этот вектор циклически вправо и разместим результат в регистре R2

$$(R_2) = X_8 X_1 X_7 X_6 X_5 X_4 X_3 X_2$$

Результат поразрядного логического И над содержимым регистров R1 и R2 разместим в регистре R3

$$(R_3) = (X_8 \wedge X_1)(X_7 \wedge X_8)(X_6 \wedge X_7)(X_5 \wedge X_6)(X_4 \wedge X_5)(X_3 \wedge X_4)(X_2 \wedge X_3)(X_2 \wedge X_1)$$

Содержимое R3 сдвинем циклически вправо на два бита и результат разместим в регистре R4

$$(R_4) = (X_2 \wedge X_3)(X_1 \wedge X_2)(X_8 \wedge X_1)(X_7 \wedge X_8)(X_6 \wedge X_7)(X_5 \wedge X_6)(X_4 \wedge X_5)(X_3 \wedge X_4)$$

Поразрядное логическое умножение содержимого регистров R3 иR4 даст

$$(R_5) = \left[(X_8 \wedge X_1) \vee (X_2 \wedge X_3)\right]\left[(X_7 \wedge X_8) \vee (X_1 \wedge X_2)\right]\left[(X_6 \wedge X_7) \vee (X_6 \wedge X_1)\right]$$

$$\left[(X_5 \wedge X_6) \vee (X_7 \wedge X_8)\right]\left[(X_4 \wedge X_5) \vee (X_6 \wedge X_7)\right]\left[(X_3 \wedge X_4) \vee (X_5 \wedge X_6)\right]$$

$$\left[(X_2 \wedge X_3) \vee (X_4 \wedge X_5)\right]\left[(X_1 \wedge X_2) \vee (X_3 \wedge X_4)\right]$$

Нетрудно видеть, что инверсия содержимого регистра R5 даст искомый результат в младших битах тетрад (символом "*" помечены не интересующие нас биты)

$$(R_5) = ***\overline{(X_8 \wedge X_7 \wedge X_6 \wedge X_5)} *** \overline{(X_1 \wedge X_2 \wedge X_3 \wedge X_4)}$$

Изложенное позволяет представить блок-схему алгоритма моделирования в виде рисунка 1.

Порядок выполнения опыта

1. Определить в качестве R1..R2 (рисунок 1) конкретные регистры процессора;

2. Написать ассемблерную программу, реализующую алгоритм (рисунок 1);

3. Ввести программу в ОЗУ и отладить ее; получить от преподавателя задание на входной вектор X, выполнить моделирование и результат показать преподавателю.

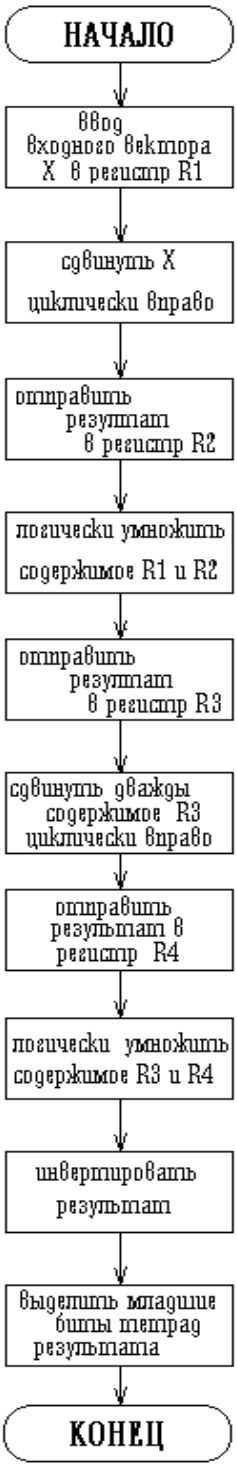


Рисунок 1 – Блок-схема алгоритма моделирования

Опыт 2. Моделирование преобразователя кодов

Программное моделирование ряда логических устройств может быть осуществлено без вычислений функций алгебры логики, как это

делалось в опыте 1. В частности, это справедливо для преобразователей кодов, программные модели которых проще всего создавать табличными методами. Рассмотрим алгоритм моделирования преобразователя двоичного кода 8-4-2-1 в двоично-десятичный 8-4-2-1. Положим, что входная информация - беззнаковая и имеет длину слова в один байт. Особенностью рассматриваемой задачи является то, что входные числа большие, чем 9910, но меньшие, чем 25610 должны представляться в двоично-десятичном коде тремя тетрадами, т.е. иметь длину в полтора байта. Поэтому, для унификации процедур выборки из таблицы целесообразно выбрать длину выходного слова преобразователя в два байта. При этом нулевые значения соответствующих тетрад в зависимости от входной информации должны соответствовать рисунку 2.

старший байт		младший байт		входное число(10)
0	0	0	0	0
0	0	0	*	1…9
0	0	*	*	10…99
0	*	*	*	100…255

Рисунок 2 – Кодирование двоично-десятичных чисел в двухбайтном формате

Порядок выполнения опыта

1. Получить от преподавателя задание на диапазон преобразуемых кодов;

2. Создать в соответствующих областях ОЗУ первую и вторую части таблицы;

3. Написать ассемблерную программу, реализующую алгоритм на рисунке 3;

4. Ввести программу в ОЗУ. Обратить внимание на необходимость ее размещения вне области 0100...02FF (занято таблицей);

5. Отладить программу, выполнить моделирование и результат показать преподавателю.

Таблица 1 – Расположение в ОЗУ двоично-десятичных кодов

Адрес (16)	Данные(16)	Адрес (16)	Данные(16)
младшие байты		старшие байты	
0100	00	0200	00
0101	01	0201	00
0102	02	:	:
:	:	0262	00
0109	09	0263	00
010A	10	0264	01
010B	11	0265	01
:	:	:	:
0162	98	02C7	01
0163	99	02C8	02
0164	00	:	:
0165	01	02CE	02
:	:	021F	02
01C6	98		
01C7	99		
01C8	00		
01C9	01		
:	:		
01FD	53	3	
01FE	54		
01FF	55		

Опыт 3. Моделирование мультиплексора К155КП5

Алгоритм работы мультиплексора К155КП5 сводится к коммутации с инвертированием одного из входов X0...X7 на выход Y в зависимости от состояния управляющих входов S0...S2. Эту модель также можно создать без непосредственного вычисления функций алгебры логики. Сущность одного из возможных алгоритмов решения рассматриваемой задачи состоит в следующем. Осуществляется циклическое сравнение вектора $S \in 0...7$ и декрементируемого счетчика циклов (СЦ). При неравенстве каждый раз происходит сдвиг вектора X влево. В момент равенства (СЦ) = S происходит выход из цикла. При этом очевидно, что нужный бит входного вектора X находится в результате сдвигов в позиции старшего бита.

Инвертирование и выделение этого бита приводит к окончательному результату. Выделение требуемого бита можно осуществить, например, выдвижением его во флаг CARRY. Соответствующая описанному алгоритму блок-схема приведена на рисунке 4.

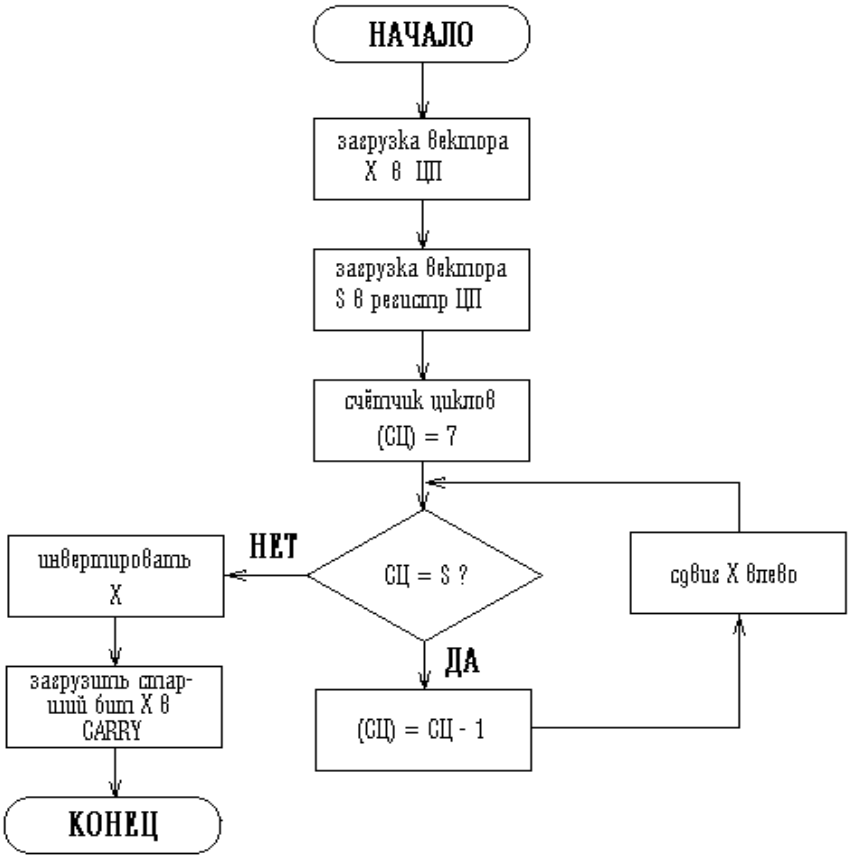


Рисунок 4 – Блок-схема алгоритма работы мультиплексора

Порядок выполнения опыта

1. Написать ассемблерную программу, реализующую алгоритм (рисунок 4);

2. Ввести программу в ОЗУ;

3. Отладить программу, выполнить моделирование, результат показать преподавателю.

Контрольные вопросы

1. Для чего используются программные модели аппаратных средств?

2. Каковы способы построения программной модели?

3. Как модифицировать алгоритм (рисунок 1) в целях автоматизации моделирования при любом значении вектора X?

4. Как оптимизировать алгоритм (рисунок 3) по объему табличной памяти?

5. Как можно представить алгоритм, альтернативный алгоритму (рисунок 3)?

6. Как из компиляционного алгоритма (рисунок 4) сделать интерпретирующий (для разных типов мультиплексоров)?

Список литературы

1. Овчаренко А.И., Программирование на ассемблере: Уч. пособие. – К.: ИСИ, 1993. -135 с.

Лабораторная работа №14

ПРОГРАММНЫЕ СРЕДСТВА БЕЙСИКА ДЛЯ ОФОРМЛЕНИЯ АССЕМБЛЕРНЫХ ВСТАВОК

Цель работы: изучение оператора POKE; функции PEEK и USR; процедур доступа к ячейкам памяти из Бейсика, входа в ассемблерные программы и выхода из них.

Опыт 1. Изучение оператора POKE

Оператор POKE X, Y используется для записи байта Y(10) в ячейку памяти с адресом X(10). Ограничением на значения адреса является верхнее значение области ОЗУ и МикроДАТ составляет X(10)<=32767. Ограничение на значение байта определено содержательным понятием байт и составляет Y(10)<=255.

Таким образом, оператор POKE позволяет с помощью программных средств Бейсика осуществить непосредственную запись данных в ячейки ОЗУ.

Порядок выполнения опыта

1. Включить питание рабочего места;
2. Запустить интерпретатор Бейсика;
3. Ввести в память машины следующую программу:

```
10 REM Запись в память
20 INPUT "Введите начальный адрес"; NA
30 DATA 98,105,116
40 FOR I=0 TO 2: READ X :POKE NA+I, X: NEXT I
50 END
```

4. Выполнить введенную программу;

5. Передать управление программе Монитор нажатием клавиши ПУСК панели контроля и отладки;

6. Проверить соответствие содержимого памяти по адресам NA, NA+1, NA+2, введенным в строке 30, числам (использовать директиву S или D). При проверке иметь в виду, что в строке 30 указаны десятичные значения, а каналы Монитора выводят шестнадцатеричные значения;

7. Результат п.6 показать преподавателю.

Опыт 2. Изучение функции PEEK

Функция PEEK X предназначена для чтения байта из ячейки памяти с адресом X(10). Ограничения на значение адреса такие же, как для оператора POKE. Таким образом, функция PEEK позволяет с помощью программных средств Бейсика осуществить непосредственное чтение ячеек ОЗУ.

Порядок выполнение опыта

1. Модифицировать программу, созданную в предыдущем опыте, путем выделения чтения в цикле ячеек памяти с адресом NA, NA+1, NA+2. Для вывода данных на экран видеотерминала использовать следующую запись: "?"; PEEK (NA+1);

2. Выполнить программу. Результат показать преподавателю;

3. Модифицировать программу по выводу информации на экран видеотерминала. Для этого заменить запись в п.2 записью "?"; CHR ¤ (PEEK (NA+1));

4. Выполнить программу. Объяснить полученное на экране сообщением. Результат показать преподавателю.

Опыт 3. Изучение функции USR и процедур входа в ассемблерные подпрограммы и выхода из них

При создании информационно-измерительных, информационно-управляющих систем, а также при отладке аппаратуры со встроенными микропроцессорами бывает необходимо реализовать часть процедур программированием в машинных кодах. Во многих версиях Бейсика для этой цели предусмотрено два способа вызова ассемблерных вставок; оператором CALL и функцией USR. В языке Бейсик-МикроДАТ используется второй способ. Оформление входа в ассемблерную подпрограмму осуществляется записью вида A=USR(ADR), где ADR – стартовый адрес подпрограммы. Входные данные для этой программы могут быть предварительно записаны в требуемые ячейки памяти с помощью операторов POKE.

Выход из-под программы осуществляется оператором RET (код C9(16)), помещенным в ее конце. Сама подпрограмма в большинстве случаев должна предусматривать запись содержимого всех регистров в стек (команды PUSH H, PUSH B, PUSH PSW) перед выполнением основных действий в подпрограмме и возврат содержимого регистров из стека (команды POP B, POP D, POP H, POP PSW) после выполнения основных действий в подпрограмме.

Порядок выполнения опыта

1. Вызвать программу Монитор;

2. Ввести в память машины программу на языке Ассемблер;

3. Выполнить введенную программу в пошаговом режиме (директива T). После выполнения команды PUSH PSW убедиться в том, что в стек записано содержимое регистров A, B, C, D, E, F, H, L. После выполнения команды IN удостоверится, что в регистр A переслано содержимое старшего байта адреса, выставленного на ПКО. После выполнения команды MOV убедиться, что содержимое регистра A переслано в ячейку 700F;

Программа на языке Ассемблер:

Адрес	Машинный код	Мнемокод
7000	C5	PUSH B
7001	D5	PUSH D
7002	E5	PUSH H
7003	F5	PUSH PSW
7004	B	IN
7005	2	
7006	1	LXI H
7007	F	
7008	0	
7009	7	MOV M,A
700A	F1	POP PSW
700B	E1	POP H
700C	D1	POP D
700D	C1	POP B
700E	6	HLT

4. Записать в ячейку 700E команду RET (код C9(16)), превращая тем самым введенную программу в подпрограмму;

5. Запустить интерпретатор Бейсика;

6. Ввести следующую программу:

```
10 INPUT "Ввод числа циклов"; N
20 FOR Y=1 TO 3
30 GOSUB 490
40 GOSUB 600
50 NEXT Y
60 STOP
490 REM п/п проверки готовности
500 FOR I=1 TO N: X=INP(Y): IF X < >0 THEN 540: NEXT I
510  ? "Наберите в порте"; Y ; "Данные,отличные от нуля"
520 ? "И вновь выполните программу"
530 STOP
540 POKE 28677, Y : USR(28672)
550 RETURN
600 REM  п/п вывода
610 ? "Байт из порта"; Y ;  "Принят"
620 X1=PEEK(28687)
630 ? "Байт равен"; X1
640 RETURN
```

7. Изобразить блок-схему алгоритма, соответствующего данной программе. По блок-схеме определить последовательность и сущность выполняемых операций. Увязать десятичные данные POKE (строка 540) и PEEK (строка 620) с параметрами подпрограммы на языке Ассемблер;

8. Выполнить программу. Результат показать преподавателю.

Содержание отчета

В отчете привести тексты всех программ, блок-схему алгоритма опыта 3.

Контрольные вопросы

1. Назначение и структура оператора POKE.
2. Назначение и структура функции PEEK.
3. Назначение и структура функции USR.
4. Как осуществить вход в ассемблерную подпрограмму?
5. Как осуществить выход из ассемблерной подпрограммы?
6. Какие действия помимо основных функций, должна выполнять подпрограмма?
7. В какой области памяти должны размещаться ассемблерные подпрограммы?
8. Почему в программе опыта 3 отсутствует операция загрузки указателя стека?
9. Как можно реализовать загрузку указателя стека в опыте 3?
10. Каково назначение строки 60 программы опыта 3?
11. Что определяет значение N, вводимое в строке 10 опыта 3?
12. Какая защита от "зацикливания" предусмотрена в программе опыта 3?

Лабораторная работа № 15

ИЗУЧЕНИЕ УЧЕБНОГО МИКРОПРОЦЕССОРНОГО КОМПЛЕКТА УМК

Цель работы: – изучить состав и технические характеристики УМК, в том числе назначение органов управления и индикации;

– изучить правила работы на УМК;

– изучить технологию ручного ввода в машинных кодах и отладки пользовательских программ.

Опыт 1. Изучение состава УМК

1.1 Назначение УМК

Учебный микропроцессорный комплект (УМК) представляет собой законченную микро-ЭВМ и предназначен для целей обучения основам построения и обслуживания микро-ЭВМ, основам программирования и отладки прикладных программ, для изучения основ сопряжения микро-ЭВМ с внешними устройствами.

1.2 Основные технические характеристики

Тип микропроцессора КР580ИК80А (КР580ВМ80А) (по ГОСТу с 1984г.)

Объём и организация ОЗУ	– 1К × 8
Объём и организация ПЗУ	– 2К × 8
в том числе ПЗУ пользователя	– 1К × 8
Возможность прерывания	– 1 вектор
Резидентное программное обеспечение	программа «Монитор»
Напряжение питания	220В ± 22В частота 50Гц ± 1Гц

1.3 Состав УМК:

– микро-ЭВМ;

– пульт оператора;

– блок питания.

Микро-ЭВМ является основной составной частью и управляет работой всего УМК. Микро-ЭВМ состоит из операционного устройства, основой которого является микропроцессор, ОЗУ, ПЗУ и устройства пошагового выполнения программ.

Пульт оператора предназначен для взаимодействия оператора с микро-ЭВМ, расположен на передней панели УМК и содержит органы, изображённые на рисунке 1.

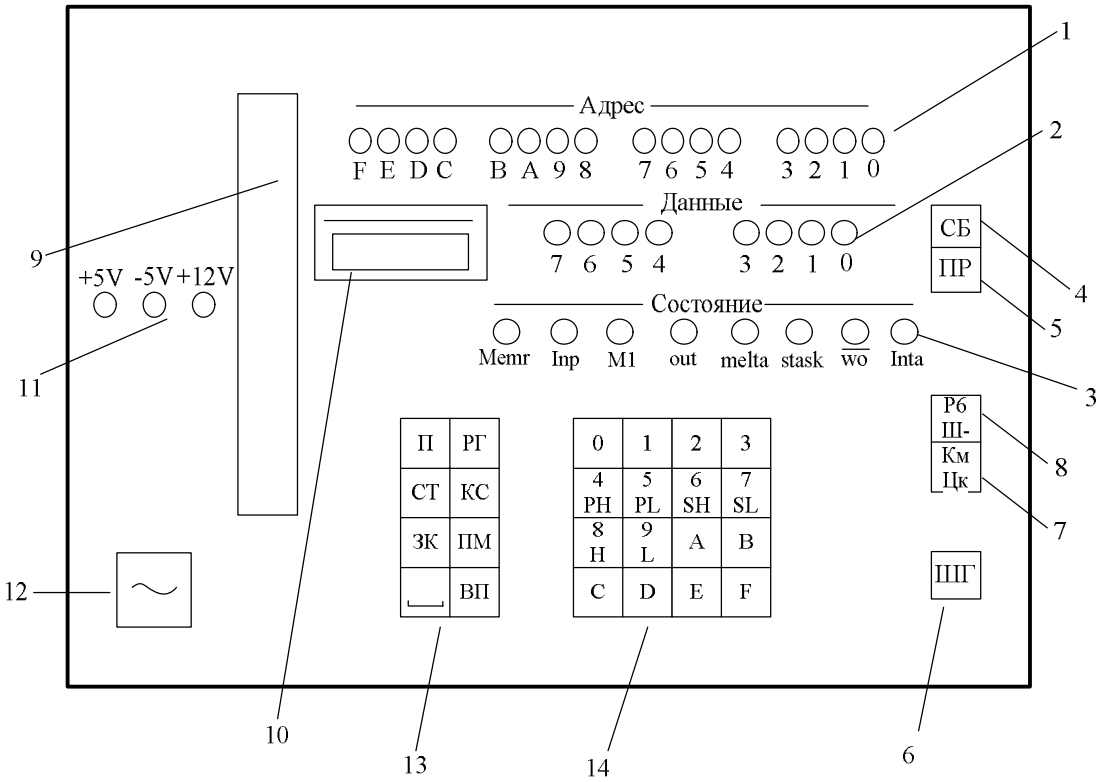


Рисунок 1 – Пульт оператора

На рисунке 1 цифрами обозначены:

1 – шестнадцатиразрядный индикатор шины адреса;

2 – восьмиразрядный индикатор шины данных;

3 – восьмиразрядный индикатор слова состояния процессора;

4 – клавиша сброса СБ;

5 – клавиша прерывания ПР;

6 – клавиша ШГ выполнения команды в пошаговом режиме;

7 – клавиша РБ\ШГ выбора режима работы;

8 – клавиша КМ\ЦК выбора величины шага;

9 – разъём для подключения макетного ТЭЗ;

10 – шестиразрядный цифро-символьный дисплей;

11 – световые индикаторы аварии по питающим напряжениям +5V, -5V, +12V;

12 – клавиша ~ подачи напряжения 220В на УМК;

13 – клавиатура директив (8 клавиш);

14 – клавиатура информации (16 клавиш);

Блок питания обеспечивает постоянными стабилизированными напряжениями (+5V, -5V, +12V) микро-ЭВМ, пульт в целом, а также макетный ТЭЗ.

1.4 Назначение органов управления и индикации

1.4.1 Клавиша ~ (позиция 12 на рисунке 1) предназначена для подключения и отключения УМК от сети 220В. Нажатое состояние клавиши соответствует подключению к сети, а отжатое - отключению.

1.4.2 Световые индикаторы (позиция 11 на рисунке 1) предназначена для индикации аварии раздельно по трём напряжениям питания +5V, -5V, +12V. Свечение индикатора означает, что соответствующий индикатор отключился от нагрузки.

1.4.3 Клавиша СБ (позиция 4 на рисунке 1) предназначена для исходной (начальной) установки элементов микро-ЭВМ. Нажатием этой клавиши осуществляется системный сброс, а в крайней левой позиции дисплея (позиция 10 на рисунке 1) должен появиться символ « _ ».

1.4.4 Клавиша ПР (позиция 5 на рисунке 1) предназначена для останова выполнения программы. При этом текущее состояние всех регистров микропроцессора сохраняется в ОЗУ, откуда они могут быть восстановлены и тем самым есть возможность продолжить программу, начиная с точки останова.

1.4.5 Клавиша РБ\ШГ (позиция 7 на рисунке 1) предназначена для выбора одного из двух возможных режимов выполнения программы: рабочего, когда следующие команды автоматически в соответствии с программой извлекаются из памяти и исполняются вслед за предыдущей, и пошагового, когда после исполнения очередной команды процессор переходит в режим ожидания и ждёт внешнего сигнала на выполнение следующей команды (шага).

1.4.6 Клавиша КМ\ЦК (позиция 8 на рисунке 1) служит для выбора одной из двух модификаций пошагового режима. При модификации КМ выполняется покомандный шаг (один шаг - одна команда), а при модификации ЦК – поцикловый шаг (один шаг - один цикл).

1.4.7 Клавиша ШГ (позиция 6 на рисунке 1) необходима для исполнения очередного шага, как в режиме КМ, так и в режиме ЦК. После нажатия клавиши ШГ и исполнения очередного шага на световых

индикаторах АДРЕС, ДАННЫЕ, СОСТОЯНИЕ отображается в двоичном коде соответственно состояние шины адреса, шины данных, и состояния процессора.

1.4.8 16-разрядный двоичный световой индикатор адреса (позиция 1 на рисунке 1) предназначена для индикации текущих адресов, выставляемых на шине адреса. Разряды индикатора занумерованы в шестнадцатеричной системе от F до 0.

A15	A14	A13	A12	A11	A10	A9	A8
2^{15}=32768	2^{14}=16384	2^{13}=8192	2^{12}=4096	2^{11}=2048	2^{10}=1024	2^{9}=512	2^{8}=256

A7	A6	A5	A4	A3	A2	A1	A0
2^{7}=128	2^{6}=64	2^{5}=32	2^{4}=16	2^{3}=8	2^{2}=4	2^{1}=2	2^{0}=1

1.4.9 8-разрядный двоичный световой индикатор данных (позиция 2 на рисунке 1) предназначен для индикации текущих данных, передаваемых по шине данных. Разряды индикатора занумерованы от 7 до 0.

D7	D6	D5	D4	D3	D2	D1	D0
2^{7}=128	2^{6}=64	2^{5}=32	2^{4}=16	2^{3}=8	2^{2}=4	2^{1}=2	2^{0}=1

1.4.10 8-разрядный двоичный световой индикатор состояния процессора (позиция 3 на рисунке 1) предназначен для индикации в начале каждого машинного цикла в двоичном коде состояния микропроцессора. Назначение разрядов слова состояния расшифровано в таблице 1.

1.4.11 6-разрядный цифро-символьный дисплей (позиция 10 на рисунке 1) предназначен для индикации в шестнадцатиразрядном коде адресов и данных памяти, состояний регистров и особых состояний (например, при неправильной работе с клавиатурой в крайней правой позиции дисплея индицируется символ «?»).

1.4.12 Клавиатура директив (позиция 13 на рисунке 1) содержит 8 клавишей, назначение которых таково:

П – чтение и изменение содержимого памяти;

РГ – чтение и изменение содержимого регистров микропроцессора;

СТ – передача управления программе пользователя;

КС – определение контрольной суммы массива памяти;

ЗК – заполнение массива константой;

ПМ – перемещение массива в адресном пространстве памяти;

«_» – пробел, служит для разделения атрибутов при вводе с клавиатуры;

ВП – выполнить директив

Таблица 1 – Назначение разрядов слова состояния

INT A	Сигнал подтверждения запроса прерывания
WO	Сигнал, указывающий на осуществление в данном машинном цикле записи в память или вывода на внешнее устройство
STACK	Сигнал, указывающий, что на шине адреса выставлено содержание указателя стека
HLT A	Сигнал, свидетельствующий об останове процессора командой HLT
OUT	Сигнал, указывающий на осуществление в данном машинном цикле вывода на внешнее устройство
M1	Сигнал, идентифицирующий машинный цикл M1 (выборка первого байта команды)
INP	Сигнал, указывающий на осуществление в данном машинном цикле операции ввода с внешних устройств
MEMR	Сигнал, указывающий на осуществление в данном машинном цикле чтения из памяти

Кодировка возможных состояний микропроцессора приведена в таблице 2.

Таблица 2 – Кодировка возможных состояний микропроцессора

Состояние МП	INTA	WO	STACK	HLTA	OUT	MI	INP	MEMR
Выбор команды	0	1	0	0	0	1	0	1
Чтение из памяти	0	1	0	0	0	0	0	1
Запись в память	0	0	0	0	0	0	0	0
Запись в стек	0	0	1	0	0	0	0	0
Ввод	0	1	0	0	0	0	1	0
Вывод	0	0	0	0	1	0	0	0
Прерывание	1	1	0	0	0	1	0	0
Прерывание в режиме «останов»	1	1	0	1	0	1	0	0
Останов	0	1	0	1	0	0	0	1

1.4.13 Клавиатура информации (позиция 14 на рисунке 1) содержит 16 клавиш для ввода чисел (F до 0). Клавиши служат для идентификации регистров микропроцессора.

1.4.14 Разъём для подключения макетного ТЭЗ предназначен для использования совместно с УМК различных внешних устройств.

1.5 Включение УМК

1.5.1 Установить кнопку «~» в отжатое состояние.

1.5.2 Подключить УМК к сети переменного тока 220В 50 Гц.

1.5.3 Клавишу РБ/ШГ установить в положение РБ.

1.5.4 Включить клавишу «~» и проконтролировать отсутствие свечения индикаторов «+5В», «-5В», «+12В». Если свечение всех трёх индикаторов отсутствует, УМК готов к работе. В противном случае необходимо выключить УМК клавишей «~» и через 30-40 сек. Осуществить повторное включение. При свечении хотя бы одного из индикаторов выключить УМК и обратиться к преподавателю.

1.5.5 Нажать клавишу СБ. При этом в крайней левой позиции дисплея должен появиться символ «–».

Опыт 2. Изучение правил работы на УМК и технологии ручного ввода и отладки программ

2.1 Индикация и изменение содержимого памяти

2.1.1 Последовательно нажмите клавиши:

П [X1 X2 X3 X4]_D1_D2 ВП,

где X1 X2 X3 X4 - шестнадцатеричный четырёхразрядный адрес ячейки памяти, к которой осуществляется обращение; доступные для пользователя ячейки памяти находятся в диапазоне адресов 800 H÷BFFH;

D1 D2 – шестнадцатеричные данные, подлежащие записи в память.

2.1.2 Для перехода к следующей ячейке памяти без изменения содержимого индицируемого после ввода адреса нажать клавишу «_» без набора данных.

2.2 Индикация и изменение содержимого регистров микропроцессора

2.2.1 Нажать клавишу РГ, а затем идентификатор регистра. Идентификаторами регистров являются следующие символы:

A – регистр A;

B – регистр B;

C – регистр C;

D – регистр D;

E – регистр E;

H – регистр H;

L – регистр L;

F – регистр условий (признаков, флагов);

SL – младший байт указателя стека;

SH – старший байт указателя стека;

PL – младший байт счётчика команд (программного счётчика);

PH – старший байт счётчика команд (программного счётчика).

Реакцией УМК на ввод идентификатора является индикация на дисплее в шестнадцатеричном коде содержимого указанного регистра.

2.2.2 Изменить содержимое выбранного в п. 2.2.1. регистра путём набора нового значения данных и последующего нажатия клавиши «_».

2.2.3 Индицировать содержимое остальных регистров без его изменения путём нажатия клавиши «_».

2.2.4 Завершить директиву нажатием клавиши ВП.

2.3 Передача управления программе пользователя

2.3.1 Написать и ввести в УМК программу. Назначение программы и её адресное пространство задаёт преподаватель.

2.3.2 Нажать последовательно клавиши:

СТ ADR1 _ ADR2 _ ADR3 ВП, где

ADR1 – начальный адрес программы;

ADR2, ADR3 – адреса останова программы, являются необязательными атрибутами командной строки (1).

При достижении программой точек ADR2, ADR3 управление передаётся программе Монитор, а содержимое регистров микропроцессора сохраняется в ОЗУ. При передаче управления из Монитора по адресу ADR1 происходит восстановление состояния процессора.

Значения ADR2, ADR3 задаёт преподаватель.

2.3.3 Проверить результат работы программы индикацией содержимого регистров (ячеек памяти). Результаты показать преподавателю.

2.4 Пошаговое выполнение программ

2.4.1 Установить клавишу РБ/ШГ в положение ШГ.

2.4.2 Установить клавишу КМ/УК в положение КМ.

2.4.3 Передать управление выполняемой программе (СТ ADR1 ВП).

2.4.4 Выполнить программу в пошаговом режиме, использую клавишу ШАГ.

2.4.5 Результаты показать преподавателю и по его указанию выключить УМК.

Содержание отчёта

Отчёт по лабораторной работе должен содержать описание назначения всех органов управления и индикации УМК.

Контрольные вопросы

1. Для чего предназначен учебный микропроцессорный комплект (УМК)?
2. Каково назначение органов управления и индикации?
3. Какова последовательность включения УМК?
4. Какое количество оперативной памяти доступно пользователю?
5. Каким образом вводится информация в ОЗУ?
6. Как можно изменить состояние регистров?
7. Какова последовательность нажатия клавиш при автоматическом выполнении введенной программы?
8. Чем отличается поцикловый режим от покомандного при пошаговом выполнении программы?

Список литературы

1. Учебный микропроцессорный комплект. Паспорт. РР.059.004.ПС.
2. Коган Б.М., Сташин В.В. Микропроцессоры в цифровых системах. М.: Энергия, 1979.
3. Конспект лекций по курсу «Основы МП – техники», часть 1.

Навчальне видання

Методичні вказівки до лабораторних робіт
з курсу «Основи мікропроцесорної техніки» (частина 2)
для студентів спеціальності 7.091302
“Метрологія та вимірювальна техніка”
денної та заочної форм навчання
Російською мовою

Укладачі: ОВЧАРЕНКО Олександр Іванович,
ЛИСЕНКО Володимир Валерійович,
МИГУЩЕНКО Руслан Павлович,
МЕДВЕДЄВА Людмила Олександрівна

Відповідальний за випуск В.І. Дякін
Роботу до друку рекомендував О.І. Рогачов

В авторській редакції

План 2003р., п. 181
Підписано до друку . Формат 60×84 1/16. Папір офсет. Друк – ризографія. Гарнітура Таймс. Умов. друк. арк. 3.2. Облік. – вид. арк. 3.5. Наклад 100 прим. Ціна договірна.

Видавничий центр НТУ “ХПІ”, 61002, Харків, вул. Фрунзе, 21
Свідоцтво про державну реєстрацію ДК №196 від 10.07.2000р. .

Друкарня НТУ “ХПІ”